

10/627,731
5

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日 2 0 0 3 年 7 月 1 0 日
Date of Application:

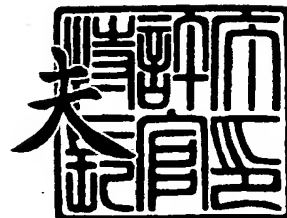
出 願 番 号 特 願 2 0 0 3 - 1 9 5 1 9 4
Application Number:
[ST. 10/C]: [J P 2 0 0 3 - 1 9 5 1 9 4]

出 願 人 株 式 会 社 リ コ ー
Applicant(s):

2 0 0 3 年 7 月 2 8 日

特 許 庁 長 官
Commissioner,
Japan Patent Office

今 井 康



出 証 番 号 出 証 特 2 0 0 3 - 3 0 5 9 7 8 5

【書類名】 特許願

【整理番号】 0305290

【提出日】 平成15年 7月10日

【あて先】 特許庁長官 太田 信一郎 殿

【国際特許分類】 G03G 21/00 370

【発明の名称】 情報処理装置およびバージョンチェック方法

【請求項の数】 25

【発明者】

 【住所又は居所】 東京都大田区中馬込 1丁目3番6号 株式会社リコー内

 【氏名】 秋吉 邦洋

【発明者】

 【住所又は居所】 東京都大田区中馬込 1丁目3番6号 株式会社リコー内

 【氏名】 田中 浩行

【発明者】

 【住所又は居所】 東京都大田区中馬込 1丁目3番6号 株式会社リコー内

 【氏名】 安藤 光男

【特許出願人】

 【識別番号】 000006747

 【氏名又は名称】 株式会社リコー

【代理人】

 【識別番号】 100070150

 【弁理士】

 【氏名又は名称】 伊東 忠彦

【先の出願に基づく優先権主張】

 【出願番号】 特願2002-224136

 【出願日】 平成14年 7月31日

【手数料の表示】

 【予納台帳番号】 002989

 【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9911477

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 情報処理装置およびバージョンチェック方法

【特許請求の範囲】

【請求項 1】 アプリケーションと、前記アプリケーションからの A P I を用いた要求に基づきシステム側の処理を行うシステムサービスとを備えた情報処理装置であって、

前記アプリケーションがシステムサービスに対して使用する A P I のバージョン情報と、当該システムサービスが有する A P I のバージョン情報を取得する取得手段と、

前記アプリケーションが前記システムサービスに対して使用する A P I のバージョンを、A P I 単位に、前記システムサービスが有する A P I のバージョンと比較する比較手段と

を有することを特徴とする情報処理装置。

【請求項 2】 前記アプリケーションが前記システムサービスに対して使用する A P I のセットのバージョンと、前記システムサービスが有する A P I のセットのバージョンとを比較する手段を更に有し、

A P I のセットのバージョンが異なっている場合にのみ、前記比較手段による比較を行う請求項 1 に記載の情報処理装置。

【請求項 3】 前記アプリケーションは、前記システムサービスに対して使用する A P I のバージョン情報を、前記アプリケーションの実行プログラム中に備え、

前記取得手段が、当該 A P I のバージョン情報を、前記アプリケーションから取得する請求項 1 に記載の情報処理装置。

【請求項 4】 前記アプリケーションから前記バージョン情報を取得するために前記アプリケーションを仮起動させる請求項 3 に記載の情報処理装置。

【請求項 5】 前記システムサービスは複数のシステムサービスを含み、前記アプリケーションは、当該複数のシステムサービスに対して使用する A P I のバージョン情報を、システムサービス毎に備え、

前記取得手段は、前記アプリケーションからあるシステムサービスに対応する

A P I のバージョン情報を取得した場合に、当該システムサービスから、当該システムサービスが有する A P I のバージョン情報を取得する請求項 3 に記載の情報処理装置。

【請求項 6】 前記アプリケーションが前記システムサービスに対して使用する A P I のバージョン情報を格納したファイルを備え、

前記取得手段が、当該 A P I のバージョン情報を、前記ファイルから取得する請求項 1 に記載の情報処理装置。

【請求項 7】 前記比較手段による比較を前記アプリケーションのインストール前に行う場合において、

前記アプリケーションが前記システムサービスに対して使用する全ての A P I のバージョンが、前記システムサービスが有する対応する A P I のバージョンと一致する場合に、当該アプリケーションがインストール可能であることを表示する請求項 1 に記載の情報処理装置。

【請求項 8】 前記情報処理装置は、前記情報処理装置におけるハードウェア資源の制御を行うコントロールサービスと、コントロールサービスをサーバとしたクライアントプロセスとして動作し、前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスとを有し、

前記システムサービスは、アプリケーションからの A P I を用いた要求を受信するコントロールサービスと、前記仮想アプリケーションサービスを含む請求項 1 ないし 7 のうちいずれか 1 項に記載の情報処理装置。

【請求項 9】 前記仮想アプリケーションサービスは前記取得手段及び前記比較手段を有する請求項 8 に記載の情報処理装置。

【請求項 10】 アプリケーションと、前記アプリケーションからの A P I を用いた要求に基づきシステム側の処理を行うシステムサービスとを備えた情報処理装置が実行する A P I のバージョンチェック方法であって、

前記アプリケーションがシステムサービスに対して使用する A P I のバージョン情報と、当該システムサービスが有する A P I のバージョン情報を取得する取得ステップと、

前記アプリケーションが前記システムサービスに対して使用する A P I のバー

ジョンを、API 単位に、前記システムサービスが有する API のバージョンと比較する比較ステップと

を有することを特徴とするバージョンチェック方法。

【請求項 11】 前記アプリケーションが前記システムサービスに対して使用する API のセットのバージョンと、前記システムサービスが有する API のセットのバージョンとを比較するステップを更に有し、

API のセットのバージョンが異なっている場合にのみ、前記比較ステップによる比較を行う請求項 10 に記載のバージョンチェック方法。

【請求項 12】 前記アプリケーションは、前記システムサービスに対して使用する API のバージョン情報を、前記アプリケーションの実行プログラム中に備え、

前記取得ステップにおいて、当該 API のバージョン情報を、前記アプリケーションから取得する請求項 10 に記載のバージョンチェック方法。

【請求項 13】 前記アプリケーションから前記バージョン情報を取得するために前記アプリケーションを仮起動させる請求項 12 に記載のバージョンチェック方法。

【請求項 14】 前記システムサービスは複数のシステムサービスを含み、前記アプリケーションは、当該複数のシステムサービスに対して使用する API のバージョン情報を、システムサービス毎に備え、

前記取得ステップにおいて、前記アプリケーションからあるシステムサービスに対応する API のバージョン情報を取得した場合に、当該システムサービスから、当該システムサービスが有する API のバージョン情報を取得する請求項 12 に記載のバージョンチェック方法。

【請求項 15】 前記アプリケーションが前記システムサービスに対して使用する API のバージョン情報を格納したファイルを備え、

前記取得ステップにおいて、当該 API のバージョン情報を、前記ファイルから取得する請求項 10 に記載のバージョンチェック方法。

【請求項 16】 前記比較ステップによる比較を前記アプリケーションのインストール前に行う場合において、

前記アプリケーションが前記システムサービスに対して使用する全てのAPIのバージョンが、前記システムサービスが有する対応するAPIのバージョンと一致する場合に、当該アプリケーションがインストール可能であることを表示する請求項10に記載のバージョンチェック方法。

【請求項17】 前記情報処理装置は、前記情報処理装置におけるハードウェア資源の制御を行うコントロールサービスと、コントロールサービスをサーバとしたクライアントプロセスとして動作し、前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスとを有し、

前記複数のシステムサービスは、アプリケーションからのAPIを用いた要求を受信するコントロールサービスと、前記仮想アプリケーションサービスを含む請求項10ないし16のうちいずれか1項に記載のバージョンチェック方法。

【請求項18】 前記仮想アプリケーションサービスが前記取得ステップ及び前記比較ステップを実行する請求項17に記載のバージョンチェック方法。

【請求項19】 アプリケーションと、前記アプリケーションからのAPIを用いた要求に基づきシステム側の処理を行うシステムサービスとを備えた情報処理装置に、

前記アプリケーションがシステムサービスに対して使用するAPIのバージョン情報と、当該システムサービスが有するAPIのバージョン情報を取得する取得手順と、

前記アプリケーションが前記システムサービスに対して使用するAPIのバージョンを、API単位に、前記システムサービスが有するAPIのバージョンと比較する比較手順と

を実行させるプログラム。

【請求項20】 前記アプリケーションが前記システムサービスに対して使用するAPIのセットのバージョンと、前記システムサービスが有するAPIのセットのバージョンとを比較する手順を更に実行させ、

APIのセットのバージョンが異なっている場合にのみ、前記比較手順を実行させる請求項19に記載のプログラム。

【請求項 2 1】 前記アプリケーションが前記システムサービスに対して使用する A P I のバージョン情報を、前記アプリケーションの実行プログラム中に備え、

前記取得手順において、当該 A P I のバージョン情報を、前記アプリケーションから取得する手順を実行させる請求項 1 9 に記載のプログラム。

【請求項 2 2】 前記アプリケーションが前記システムサービスに対して使用する A P I のバージョン情報を格納したファイルを備え、

前記取得手順において、当該 A P I のバージョン情報を、前記ファイルから取得する手順を実行させる請求項 1 9 に記載のプログラム。

【請求項 2 3】 前記比較手順における比較を前記アプリケーションのインストール前に行う場合において、

前記アプリケーションが前記システムサービスに対して使用する全ての A P I のバージョンが、前記システムサービスが有する対応する A P I のバージョンと一致する場合に、当該アプリケーションがインストール可能であることを表示する手順を実行させる請求項 1 9 に記載のプログラム。

【請求項 2 4】 アプリケーションからの要求に基づきシステム側の処理を行うシステムサービスを備えた情報処理装置で実行される当該アプリケーションのプログラムであって、情報処理装置に、

前記システムサービスからの要求に基づき、仮起動するか通常起動するかを判断する手順と、

仮起動の場合に、前記システムサービスと通信することにより、アプリケーションに関する情報を前記システムサービスに提供する手順と

を実行させるプログラム。

【請求項 2 5】 請求項 1 9 ないし 2 4 のうちいずれか 1 項に記載のプログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

この発明は、情報処理装置で実行されるアプリケーションと、情報処理装置の

システム間でのバージョン不整合による不具合発生を未然に防止することを可能とする技術に関する。

【0002】

【従来の技術】

近年では、情報処理装置の1つとして、プリンタ、コピー、ファクシミリ、スキャナなどの各装置の機能を1つの筐体内に収納した画像形成装置（以下、「複合機」という。）が知られている。この複合機は、1つの筐体内に表示部、印刷部および撮像部などを設けるとともに、プリンタ、コピーおよびファクシミリ装置にそれぞれ対応した3種類のソフトウェアを設け、これらのソフトウェアを切り替えることによって、当該装置をプリンタ、コピー、スキャナまたはファクシミリ装置として動作させるものである。

【0003】

このような従来の複合機では、プリンタ、コピー、ファクシミリ、スキャナなどの各機能単位でアプリケーションプログラムが起動され、ハードウェア資源にアクセスする機能を持ち合わせている。その際、アプリケーションプログラムが前提とするオペレーティングシステム（OS）と、実際のオペレーティングシステム（OS）のバージョンが同じことが前提となるが、例えば、OSがバージョンアップして、アプリケーションとの間でバージョン差が生じた場合、今まで使えていた機能が使えなくなったり、アプリケーションそのものが起動しなくなったりすることがある。

【0004】

このため、従来の複合機では、OSがバージョンアップされた場合、それに伴ってアプリケーションをコンパイルし直すなどして、常に整合性のとれたバージョン関係にあることが要請されている。

【0005】

【特許文献1】

特開 2002-82806 号公報

【0006】

【発明が解決しようとする課題】

ところで、このような従来の複合機では、プリンタ、コピー、スキャナおよびファクシミリ装置に対応するソフトウェアをそれぞれ別個に設けているため、各ソフトウェアの開発に多大の時間を要する。このため、出願人は、表示部、印刷部および撮像部などの画像形成処理で使用されるハードウェア資源を有し、プリンタ、コピーまたはファクシミリなどの各ユーザサービスにそれぞれ固有の処理を行うアプリケーションを複数搭載し、これらのアプリケーションとハードウェア資源との間に介在して、ユーザサービスを提供する際に、アプリケーションの少なくとも2つが共通的に必要とするハードウェア資源の管理、実行制御並びに画像形成処理を行う各種コントロールサービスからなるプラットフォームを備えた画像形成装置（複合機）を発明した。

【0007】

このような新規な複合機では、アプリケーションとコントロールサービスが別個に設けられているため、複合機の出荷後にユーザもしくは第三者であるサードベンダが新規なアプリケーションを開発して複合機に搭載することが可能であり、これによって多種多様な機能を提供することが可能となっている。

【0008】

このような新規な複合機では、アプリケーションの少なくとも2つが共通的に必要とするサービスを提供するコントロールサービスがアプリケーションと別個に設けられているので、新規アプリケーションを開発する場合、各種コントロールサービスとのプロセス間通信を実現する処理をソースコードで記述する必要がある。

【0009】

そして、新規アプリケーションを開発する場合は、各コントロールサービスが提供するアプリケーションプログラムインタフェース（API：関数、イベントを含む）を正確に把握した上で開発しなければならない。しかし、各コントロールサービスのアプリケーションプログラムインタフェース（API）がデバッグや機能追加によってバージョンアップが繰り返されると、ベンダーはどのバージョンに合わせてアプリケーション開発をすればよいかが非常に分かり難くなり、アプリが使用するAPIのバージョンとコントロールサービスにおけるAPIの

バージョンとが異なる場合も生じ得る。この場合にアプリを実行するとエラーが発生し、複合機のシステムに影響を及ぼす恐れがある。

【0010】

このことは、固定的な複数の機能を寄せ集めた従来の複合機では問題にならなかった新規な課題である。また、上記コントロールサービス以外にも、アプリケーションにAPIを用いてサービスを提供するモジュールが導入された場合には、同様の問題がある。更に、上記の問題点は画像形成装置について説明したが、この問題点はシステムソフトウェア上でアプリケーションを動作させる一般的な情報処理装置（例えば、一般的なコンピュータ）に共通の問題である。

【0011】

この発明は上記に鑑みてなされたもので、コントロールサービスなどのシステム側のサービスにおけるAPIのバージョンと、アプリケーションが使用するAPIのバージョンの違いをチェックする技術を提供することを目的とする。

【0012】

【課題を解決するための手段】

上記目的を達成するため、請求項1に記載の発明は、アプリケーションと、前記アプリケーションからのAPIを用いた要求に基づきシステム側の処理を行うシステムサービスとを備えた情報処理装置であって、前記アプリケーションがシステムサービスに対して使用するAPIのバージョン情報と、当該システムサービスが有するAPIのバージョン情報を取得する取得手段と、前記アプリケーションが前記システムサービスに対して使用するAPIのバージョンを、API単位に、前記システムサービスが有するAPIのバージョンと比較する比較手段とを有する。

【0013】

本発明によれば、アプリが使用するAPI単位毎のバージョン情報とシステムサービスのAPI単位毎のバージョン情報を取得し、対応するAPI同士のバージョンが同じか否かをチェックする。API単位にチェックするので、例えば、アプリが使用していないシステムサービス側のAPIが変更、追加、あるいは削除されて、システムサービスのAPI全体のバージョンが変わっても、アプリの

動作に影響を与えないことを判断できる。

【0014】

請求項2に記載の発明は、請求項1の記載において、前記アプリケーションが前記システムサービスに対して使用するAPIのセットのバージョンと、前記システムサービスが有するAPIのセットのバージョンとを比較する手段を更に有し、APIのセットのバージョンが異なっている場合にのみ、前記比較手段による比較を行う。

【0015】

本発明によれば、APIのセットのバージョンが異なっている場合にのみAPI単位の比較を行うので、効率良くバージョンチェックを行うことができる。

【0016】

請求項3に記載の発明は、請求項1の記載において、前記アプリケーションは、前記システムサービスに対して使用するAPIのバージョン情報を、前記アプリケーションの実行プログラム中に備え、前記取得手段が、当該APIのバージョン情報を、前記アプリケーションから取得する取得する。

【0017】

本発明によれば、APIのバージョン情報を、アプリケーションの実行プログラム中に備えるので、アプリケーションからバージョン情報を取得することができる。

【0018】

請求項4に記載の発明は、請求項3の記載において、前記アプリケーションから前記バージョン情報を取得するために前記アプリケーションを仮起動させるものである。本発明によれば、通常起動をする場合と比較して、情報処理装置のリソースを実質的に確保することがないので効率的にバージョンチェックすることができる。また、通常起動をすることなくバージョンチェックをできるので、通常起動することにより情報処理装置に影響を及ぼす恐れがなくなる。

【0019】

請求項5に記載の発明は、請求項3の記載において、前記システムサービスは複数のシステムサービスを含み、前記アプリケーションは、当該複数のシステム

サービスに対して使用するAPIのバージョン情報を、システムサービス毎に備え、前記取得手段は、前記アプリケーションからあるシステムサービスに対応するAPIのバージョン情報を取得した場合に、当該システムサービスから、当該システムサービスが有するAPIのバージョン情報を取得する。

【0020】

本発明によれば、アプリが使用する複数のシステムサービスに対してAPIのバージョンチェックを行うことが可能となる。

【0021】

請求項6に記載の発明は、請求項1に記載において、前記アプリケーションが前記システムサービスに対して使用するAPIのバージョン情報を格納したファイルを備え、前記取得手段が、当該APIのバージョン情報を、前記ファイルから取得する。本発明によれば、バージョン情報を記録したファイルを情報処理装置に格納しておくことによりバージョンチェックが可能となる。

【0022】

請求項7に記載の発明は、請求項1に記載において、前記比較手段による比較を前記アプリケーションのインストール前に行う場合において、前記アプリケーションが前記システムサービスに対して使用する全てのAPIのバージョンが、前記システムサービスが有する対応するAPIのバージョンと一致する場合に、当該アプリケーションがインストール可能であることを表示する。

【0023】

本発明によれば、バージョンチェックの結果、どのアプリケーションをインストールしてよいかを確認することができる。

【0024】

請求項8に記載の発明は、請求項1ないし7のうちいずれか1項の記載において、前記情報処理装置は、前記情報処理装置におけるハードウェア資源の制御を行うコントロールサービスと、コントロールサービスをサーバとしたクライアントプロセスとして動作し、前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスとを有し、前記システムサービスは、アプリケーションからのAPIを用いた要求を受信するコントロール

サービスと、前記仮想アプリケーションサービスを含む。

【0025】

また、請求項9に記載の発明は、請求項8の記載において、前記仮想アプリケーションサービスは前記取得手段及び前記比較手段を有するものである。

【0026】

請求項10～18に記載の発明は、上記の情報処理装置が実行するバージョンチェック方法の発明であり、請求項19～23に記載の発明は、情報処理装置にバージョンチェックの処理を実行させるプログラムの発明である。

【0027】

請求項24に記載の発明は、アプリケーションからの要求に基づきシステム側の処理を行うシステムサービスを備えた情報処理装置で実行される当該アプリケーションのプログラムであって、情報処理装置に、前記システムサービスからの要求に基づき、仮起動するか通常起動するかを判断する手順と、仮起動の場合に、前記システムサービスと通信することにより、アプリケーションに関する情報を前記システムサービスに提供する手順とを実行させるプログラムである。

【0028】

本発明によれば、仮起動をすることができるアプリケーションを提供することが可能となる。

【0029】

請求項25に記載の発明は、上記のプログラムを記録したコンピュータ読み取り可能な記録媒体である。

【0030】

【発明の実施の形態】

以下に添付図面を参照して、この発明の実施の形態にかかる情報処理装置の好適な実施の形態を詳細に説明する。なお、本発明の実施の形態では、情報処理装置の一例として画像形成装置を取り上げて説明する。

【0031】

(実施の形態1)

図1は、この発明の実施の形態1である画像形成装置（以下、「複合機」とい

う)の構成を示すブロック図である。図1に示すように、複合機100は、白黒レーザプリンタ(B&W LP)101と、カラーレーザプリンタ(Color LP)102と、スキャナ、ファクシミリ、ハードディスク、メモリ、ネットワークインタフェースなどのハードウェアリソース103を有するとともに、プラットフォーム120とアプリケーション130と仮想アプリケーションサービス(VAS: Virtual Application Service)140から構成されるソフトウェア群110とを備えている。

【0032】

仮想アプリケーションサービス(VAS)140は、アプリケーション130とプラットフォーム120の間に配置される。VAS140は、アプリケーション(以下、アプリともいう)130の各アプリが初めて登録されるときに、同時に登録処理が行われ、アプリから見るとプラットフォーム120のサービス層として認識され、サービス層から見るとアプリとして認識されるように登録される。すなわち、VAS140は、コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつアプリケーションをクライアントとしたサーバプロセスとして動作するものである。

【0033】

このVAS140はその基本機能としてラッピング機能を備えている。この機能により、アプリ130とプラットフォーム120との間のバージョン差を吸収すると共に、コントロールサービスからのメッセージを取捨選択してプラットフォーム120を意図的に隠蔽することができる。また、アプリ130とコントロールサービスとの間で隠蔽すべきAPIを隠蔽し、特定のAPIだけを開示して重要なAPIの秘匿性を確保することができる。

【0034】

しかし、バージョン差を吸収するラッピング機能があったとしても、VAS140についてはコントロールサービス側と整合性がとれるようにバージョンアップが行われるので、アプリケーションとVAS140との間でバージョンの不整合が生じる恐れがあり、その場合、実行時におけるエラーが発生する。

【0035】

そこで、実行時におけるエラー発生を防止するため、アプリと VAS 140 の全体バージョン同士を比較して、バージョンが一致しているか否かをチェックする機能を有している。

【0036】

なお、ここで、VAS 140 の全体バージョンとは、VAS 140 が提供する API のセットのバージョンであり、アプリの全体バージョンとは、アプリが VAS 140 に対して使用する API のセットのバージョンである。

【0037】

また、アプリが VAS 140 に対して使用する API 毎のバージョンをチェックする機能も有している。この機能を有することにより、アプリと VAS の全体バージョンが違っていても、各 API 同士を比較することにより、アプリケーションが動作できると判断した場合には、バージョンの整合性がとれていると判断することができる。

【0038】

プラットフォーム 120 は、アプリケーションからの処理要求を解釈してハードウェア資源の獲得要求を発生させるコントロールサービスと、一または複数のハードウェア資源の管理を行い、コントロールサービスからの獲得要求を調停するシステムリソースマネージャ (SRM) 123 と、汎用 OS 121 とを有する。

【0039】

コントロールサービスは、複数のサービスモジュールから形成され、SCS (システムコントロールサービス) 122 と、ECS (エンジンコントロールサービス) 124 と、MCS (メモリコントロールサービス) 125 と、OCS (オペレーションパネルコントロールサービス) 126 と、FCS (ファックスコントロールサービス) 127 と、NCS (ネットワークコントロールサービス) 128 とから構成される。

【0040】

また、このプラットフォーム 120 における各サービスは、前記アプリケーション 130 からの処理要求を受信し、所定のサービスをアプリケーションに提供するためのアプリケーションプログラムインタフェース (API) を有している。

なお、本明細書では、上記サービスを提供するための関数の他、アプリケーションからプラットフォーム120側に通知されるイベント、プラットフォーム120側からアプリケーションに通知されるイベントを含めてAPIと称することとする。

【0041】

汎用OS121は、UNIX（登録商標）などの汎用オペレーティングシステムであり、プラットフォーム120並びにアプリケーション130の各ソフトウェアをそれぞれプロセスとして並列実行する。

【0042】

SRM123のプロセスは、SCS122とともにシステムの制御およびリソースの管理を行うものである。SRM123のプロセスは、スキャナ部やプリンタ部などのエンジン、メモリ、HDDファイル、ホストI/O（セントロI/F、ネットワークI/F、IEEE1394 I/F、RS232C I/Fなど）のハードウェア資源を利用する上位層からの要求にしたがって調停を行い、実行を制御する。

【0043】

具体的には、このSRM123は、要求されたハードウェア資源の利用が可能であるか（他の要求により利用されていないかどうか）を判断し、利用可能であれば要求されたハードウェア資源が利用可能である旨を上位層に伝える。また、SRM123は、上位層からの要求に対してハードウェア資源の利用スケジューリングを行い、要求内容（例えば、プリンタエンジンにより紙搬送と作像動作、メモリ確保、ファイル生成など）を直接実施している。

【0044】

SCS122のプロセスは、アプリ管理、操作部制御、システム画面表示、LED表示、リソース管理、割り込みアプリ制御などを行う。

【0045】

ECS124のプロセスは、白黒レーザプリンタ（B&W LP）101、カラーレーザプリンタColor LP）102、スキャナ、ファクシミリなどからなるハードウェアリソース103のエンジンの制御を行う。

【0046】

MCS125のプロセスは、画像メモリの取得および解放、ハードディスク装置（HDD）の利用、画像データの圧縮および伸張などを行う。

【0047】

FCS127のプロセスは、システムコントローラの各アプリ層からPSTN／ISDN網を利用したファクシミリ送受信、BKM（バックアップSRAM）で管理されている各種ファクシミリデータの登録／引用、ファクシミリ読みとり、ファクシミリ受信印刷、融合送受信を行うためのAPIを提供する。

【0048】

NCS128のプロセスは、ネットワークI/Oを必要とするアプリケーションに対して共通に利用できるサービスを提供するためのプロセスであり、ネットワーク側から各プロトコルによって受信したデータを各アプリケーションに振り分けたり、アプリケーションからデータをネットワーク側に送信する際の仲介を行う。具体的には、ftpd, httpd, lpd, snmpd, telnetd, smtpdなどのサーバデーモンや、同プロトコルのクライアント機能などを有している。

【0049】

OCS126のプロセスは、オペレータ（ユーザ）と本体間の情報伝達手段となるオペレーションパネル（操作パネル）の制御を行う。OCS126は、オペレーションパネルからキー押下をキーイベントとして取得し、取得したキーに対応したキーイベント関数をSCS122に送信するOCSプロセスの部分と、アプリケーション130またはコントロールサービスからの要求によりオペレーションパネルに各種画面を描画出力する描画関数やその他オペレーションパネルに対する制御を行う関数などが予め登録されたOCSライブラリの部分とから構成される。このOCSライブラリは、アプリケーション130およびコントロールサービスの各モジュールにリンクされて実装されている。なお、OCS126のすべてをプロセスとして動作させるように構成しても良く、あるいはOCS126のすべてをOCSライブラリとして構成しても良い。

【0050】

アプリケーション130は、ページ記述言語（PDL）、PCLおよびポスト

スクリプト (PS) を有するプリンタ用のアプリケーションであるプリンタアプリ 111 と、コピー用アプリケーションであるコピーアプリ 112 と、ファクシミリ用アプリケーションであるファックスアプリ 113 と、スキャナ用アプリケーションであるスキャナアプリ 114 と、ネットワークファイル用アプリケーションであるネットファイルアプリ 115 と、工程検査用アプリケーションである工程検査アプリ 116 とを有している。これらの各アプリは、その起動時に VAS 140 に対して自プロセスのプロセス ID とともにアプリ登録要求メッセージを送信し、アプリ登録要求メッセージを受信した VAS 140 によって、起動したアプリに対する登録が行われるようになっている。

【0051】

アプリケーション 130 の各プロセス、コントロールサービスの各プロセスは、関数呼び出しとその戻り値送信およびメッセージの送受信によってプロセス間通信を行いながら、コピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるユーザサービスを実現している。

【0052】

このように、実施の形態 1 にかかる複合機 100 には、複数のアプリケーション 130 および複数のコントロールサービスが存在し、いずれもプロセスとして動作している。そして、これらの各プロセス内部には、一または複数のスレッドが生成されて、スレッド単位の並列実行が行われる。そして、コントロールサービスがアプリケーション 130 に対し共通サービスを提供しており、このため、これらの多数のプロセスが並列動作、およびスレッドの並列動作を行って互いにプロセス間通信を行って協調動作をしながら、コピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるユーザサービスを提供するようになっている。また、複合機 100 には、サードベンダなどの第三者がコントロールサービス層の上のアプリケーション層に新規アプリ 117、118 を開発して搭載することが可能となっている。図 1 では、この新規アプリ 117、118 を搭載した例を示している。

【0053】

なお、実施の形態 1 にかかる複合機 100 では、複数のアプリケーション 13

0のプロセスと複数のコントロールサービスのプロセスとが動作しているが、アプリケーション130とコントロールサービスのプロセスをそれぞれ単一の構成とすることも可能である。また、各アプリケーション130は、アプリケーションごとに追加または削除することができる。

【0054】

図2に複合機100のハードウェア構成例を示す。

【0055】

複合機100は、コントローラ160と、オペレーションパネル175と、ファックスコントロールユニット(FCU)176と、プリンタ等の画像形成処理に特有のハードウェア資源であるエンジン部177とを含む。コントローラ160は、CPU161と、システムメモリ162と、ノースブリッジ(NB)163と、サウスブリッジ(SB)164と、ASIC166と、ローカルメモリ167と、HDD168と、ネットワークインターフェースカード(NIC)169と、SDカード用スロット170と、USBデバイス171と、IEEE1394デバイス172と、センタロニクス173とを含む。なお、メモリ162、167はRAM、ROM等を含む。FCU176およびエンジン部177は、コントローラ160のASIC166にPCIバス178で接続されている。

【0056】

CPU161が、複合機100にインストールされるアプリケーション、コントロールサービス等のプログラムを、メモリから読み出して実行する。

【0057】

図3は、実施の形態1にかかる複合機100のVAS140の構成と、VAS140と各アプリ、コントロールサービス層150および汎用OS121との関係を示すブロック図である。なお、図3では、アプリケーション130の例として、プリンタアプリ111、コピーアプリ112、新規アプリ117、118を示しているが、他のアプリでも同様の構成である。

【0058】

仮想アプリケーションサービス(VAS)140のプロセスには、ディスパッチャ145と、制御スレッド144と、バージョン情報取得スレッド143と、

全体バージョンチェックスレッド142と、API単位バージョンチェックスレッド141とが動作している。また、アプリは使用APIテーブル212を有し、VAS140は全APIテーブル211を有している。なお、アプリケーションの起動（仮起動含む）時、VAS140の起動時には、使用APIテーブル212、全APIテーブル211はそれぞれ、例えば、RAM210に展開される。

【0059】

ディスパッチャ145は、アプリケーション130やコントロールサービスからのメッセージ受信を監視し、受信したメッセージに応じて制御スレッド144、バージョン情報取得スレッド143、全体バージョンチェックスレッド142、API単位バージョンチェックスレッド141に処理要求を行うものである。実施の形態1の複合機100では、ディスパッチャ145により、仮起動もしくは通常起動したアプリとVAS140との間で行われるプロセス間通信を行うことができる。なお、仮起動とは、アプリケーションとVASとが通信をするためだけにアプリケーションを仮に起動することであり、詳細については後述する。

【0060】

制御スレッド144は、ディスパッチャ144を介して送られてくる各アプリが使用するAPI単位毎のバージョン情報や各アプリの全体バージョン情報などを、全体バージョンチェックスレッド142、API単位バージョンチェックスレッド141にそれぞれ渡したり、各スレッド141～143間における処理順序を制御したり、各スレッド141～143からの処理要求をディスパッチャ145に伝えたりする。

【0061】

また、API単位バージョンチェックスレッド141は、アプリがVAS140に対して使用するAPI単位毎のバージョン情報と、VAS140の全API単位毎のバージョン情報とを取得して、対応するAPI同士のバージョンを比較して、同じか否かをチェックするものである。

【0062】

また、全体バージョンチェックスレッド142は、各アプリ毎の全体バージョン情報と、VAS140の全体バージョン情報とを取得して、全体バージョン同士が同じか否かをチェックするものである。このスレッドの機能を有することにより、まず全体バージョン同士を比較して、全体バージョン同士が同じであれば整合性が有りと言えるので、API単位毎のバージョンチェックを行う必要がなくなり、処理を簡略化することができる。そして、API単位毎のバージョンチェックは、全体バージョンが違っている場合にのみ行えばよい。

【0063】

バージョン情報取得スレッド143は、API単位バージョンチェックスレッド141で必要なアプリが使用するAPI単位毎のバージョン情報とVASのAPI単位毎のバージョン情報を取得したり、全体バージョンチェックスレッド142で必要なアプリの全体バージョン情報とVASの全体バージョン情報を取得するものである。

【0064】

本実施の形態では、新規アプリのインストール時にバージョンチェックを行うが、まだアプリが起動されていないためプロセス間通信を使って情報を取得することができない。従って、バージョン情報取得スレッド143は、対象となるアプリを仮起動させて、プロセス間通信によりアプリから必要な情報を取得する。なお、アプリから取得する情報として、API単位毎のバージョン情報以外にもプロダクトID（ベンダー、アプリ、バージョンから一義的に決定される）、ベンダー名、アプリケーション名、全体バージョン、リソース情報等を取得することも可能である。

【0065】

なお、図3のように複数のスレッド141～144を用いる構成とする代わりに、スレッド141～144を1つのスレッドとして構成してもよい。その場合、その1つのスレッドが、本実施の形態におけるバージョンチェックの一連の処理（バージョン情報取得、バージョンチェック）を実行する。なお、VAS140のプログラムは、SDカード、CD-ROM等の記録媒体に格納して配布できる。また、ネットワークを介して配布することもできる。また、このプログラム

は、例えば、SDカードを複合機に挿入することにより複合機にインストールすることができる。また、SDカードから起動することもできる。

【0066】

図4は、本発明のバージョンチェックの方法の概要を説明するための図である。図4に示すように、アプリケーションの実行ファイルに当該アプリケーションがVAS140に対して使用するAPIのバージョン情報（バージョンテーブル）を含めておく。同様に、VAS140の実行ファイルにも、VASが有しているAPIのバージョン情報（バージョンテーブル）を含めておく。なお、一般に、アプリケーションがVAS140に対して使用する複数のAPIは、VAS140が有している複数のAPIの一部である。

【0067】

VAS140が有するAPIのバージョン情報は、API毎のバージョンと、APIのセットとしてのバージョンである。また、アプリケーションが有するバージョン情報は、API毎のバージョンと、APIのセットとしてのバージョンを有している。アプリケーションが使用するAPIセットのバージョンは、アプリケーションを作成するときに使用したVASのAPIセットのバージョンと同じである。

【0068】

実行ファイルにバージョン情報を含めるには、例えば、図4に示すように、バージョン情報をインクルードファイルとして作成しておき、作成したプログラムにインクルードしてコンパイルする。

【0069】

VAS140は、仮起動されたアプリからプロセス間通信によりアプリが使用するAPIのバージョン情報を取得し、当該バージョン情報と、VASが有するAPIのバージョン情報とをAPI毎に比較して、アプリケーションがインストール可能かどうかを判断する。また、APIセットのバージョン同士を比較し、これらが異なる場合にのみAPI毎のバージョン比較をしてもよい。なお、VAS140は、アプリが使用するバージョン情報のテーブル全体を取得して比較処理を行ってもよいし、アプリが使用するAPIのバージョン情報をAPI毎に1

つずつ取得して対応するAPIバージョンとの比較を行ってもよい。

【0070】

更に、上記のようにバージョン情報を実行ファイルに含める代わりに、アプリのバージョン情報を別ファイルとして複合機100に格納しておき、VAS140がそのファイルを参照するようにしてもよい。

【0071】

バージョン情報はVASが参照できさえすればよく、バージョン情報を複合機100内に保持する方法は上記の方法以外の種々の方法を用いることができる。

【0072】

さて、アプリが作成された後に、VAS140のAPIセットのバージョンが上がり、アプリが使用するAPIセットのバージョンと、VASが有するAPIセットのバージョンとが異なる場合が起こり得る。

【0073】

例えば、図5の例では、API No. = 251のバージョンが(101) → (102)にバージョンアップしたために、VAS140のAPIセットのバージョンが(1.00) → (1.01)となっている。

【0074】

この場合、APIセットを比較しただけでは、バージョンが異なるので、アプリとVASとの整合性がないと判断され得るが、実際には、API No. 251はアプリで使用しないAPIであるため、アプリは問題なく動作できる。従って、API毎の比較を行うことにより、APIセット同士のバージョンが異なっても、アプリが問題なく動作できると判断できる。

【0075】

次に、アプリケーションの仮起動について説明する。仮起動は、複合機のリソースを使用することになるアプリの通常起動（アプリ本来の機能を奏するための起動を通常起動と呼ぶ）とは別の起動である。仮起動では、アプリはアプリ本来の動作に必要なメモリ確保等のリソース取得を行わず、VAS140とのプロセス間通信処理のみを行う。そして、アプリは、VAS140がバージョンチェックを含むアプリに関するチェックを行うために必要な情報をVAS140に提供

する。仮起動したアプリのプロセスは、V A S 1 4 0 との通信処理が終了すれば終了する。また、アプリの仮起動に関する機能は、アプリ本来の機能によらず、本実施の形態における複合機 1 0 0 で動作するアプリに共通する機能である。従って、例えば、ベンダーがアプリを開発する場合、ベンダーに、仮起動の機能を含むプログラムテンプレートを提供し、そのプログラムテンプレートを用いてベンダーが複合機用のアプリを開発することができる。なお、上記のバージョンチェックを実現するためには、例えば、ベンダーが、アプリ開発に使用した A P I とそのバージョンを記録したインクルードファイルを作成し、アプリのコンパイル時にインクルードする。

【0076】

アプリが仮起動の機能を持つことにより、通常起動をすることなく V A S との通信により V A S にアプリ情報を提供でき、V A S がアプリのチェックを行うことができる。従って、A P I バージョン等の整合性がとれていないアプリを通常起動することによりアプリが異常動作し、複合機に影響を与える恐れがなくなる。

【0077】

図 6 に、仮起動の機能を含むアプリのプログラム記述（メイン関数）の概要を示す。なお、この記述を上記のプログラムテンプレートとして提供する。

【0078】

図 6 に示すように、このプログラム記述は、アプリケーションを仮起動するか通常起動するかを引数（-v）によって指定する。これにより、V A S がアプリを起動する際に、通常起動と仮起動とを容易に使い分けることができる。すなわち、引数（-v）を使って仮起動を指定すると、仮起動が実行され、アプリ情報提供処理がなされる。また、仮起動が指定されていない場合は、通常起動を行って、アプリ本来の動作を行う。

【0079】

なお、後述するアプリ起動時、あるいは、アプリ起動後の実行時にバージョンチェックする場合は、この通常起動処理が選択される。

【0080】

このように仮起動の機能を有するアプリのプログラムは、SDカード、CD-ROM等の記録媒体に格納して配布できる。また、ネットワークを介して配布することもできる。また、このプログラムは、例えば、SDカードを複合機に挿入することにより複合機にインストールすることができる。また、SDカードから起動することもできる。

【0081】

次に、アプリインストール時にバージョンチェックを行う場合の複合機の動作について、図7のフローチャートを用いて説明する。図8は、図7の中のバージョンチェック処理のサブルーチンを示すフローチャートである。

【0082】

まず、図7に示すように、ユーザがオペレーションパネル上のキーやボタンを使ってアプリのインストール処理を選択すると（ステップS501）、SCS（システムコントロールサービス）122からVAS140にアプリインストール処理の開始を要求し（ステップS502）、VAS140がインストール対象エリア内のアプリを仮起動させる（ステップS503）。この仮起動は、前述したように、インストールするアプリ内の各種情報をプロセス間通信を使ってVAS140に通知し、VASがバージョンチェック処理を行うためのものである（ステップS504）。

【0083】

ステップS504におけるバージョンチェック処理の詳細を図8を用いて説明する。

【0084】

図8に示すように、アプリから通知された情報にアプリの全体バージョン情報（APIセットのバージョン情報）が含まれているか否かを判断し（ステップS601）、全体バージョン情報が含まれている場合は、VAS140のAPIの全体バージョンと比較し、同じであれば（ステップS602）、バージョンの不整合はなく、VAS140は当該アプリケーションに対して動作保証する（ステップS603）。なお、動作保証するとは、例えば、当該アプリに対応づけてAPIバージョンに関して問題ない旨のフラグを記録しておくことである。

【0085】

また、図8のステップS601もしくはステップS602において、NOであった場合には、ステップS604に移行し、アプリが使用するAPIのバージョンと、それに対応したVAS140のAPIのバージョンとが、アプリが使用するAPI毎に1つ1つ比較され、全て同じバージョンであるか否かが判断され、同じであれば上記ステップS603にてVAS140のサポート範囲内として動作保証する（ステップS603）。しかしステップS604でバージョンの異なるAPIがあった場合は、VAS140のサポート範囲外となるため、その旨を記録する（ステップS605）。

【0086】

このように、図8のステップS504のサブルーチンでバージョンチェック処理を行った後、他にインストール処理を行うアプリが無くなるまで上記ステップS503に戻り、上記バージョンチェック処理が繰り返される（ステップS505）。

【0087】

インストール処理するアプリが無くなった場合は、ステップS506でインストール時に取得したアプリの情報に基づいてインストール画面をオペレーションパネル上などに生成し、各種情報を表示する（ステップS506）。例えば、インストール処理された複数のアプリのバージョンチェック結果に基づいて動作保証の有無を表示する。これにより、ユーザは、正式なインストール要求すべきアプリをインストール画面から容易に選択することができ（ステップS507）、選択されたアプリのみを最終的にインストール処理する（ステップS508）。

【0088】

上記したように、実施の形態1では、アプリのインストール時に、VASによってアプリを仮起動させ、プロセス間通信を利用してバージョン情報を含むアプリに関する情報を取得して、バージョンチェック行うことができるので、動作保証されたアプリのみを選択的にインストール処理することができる。

【0089】

また、インストール時の他、アプリの通常起動時に本発明のバージョンチェッ

ク処理を行うことができる。アプリを仮起動させるか通常起動させるかは、図6のメイン関数記述例に示したように、引数（-v）を用いるか否かで決まり、通常起動時にも勿論プロセス間通信を使ってアプリのバージョン情報などが取得できるため、バージョンチェック処理を行うことができる。このアプリの起動時におけるバージョンチェックは、既にインストールされているアプリを初めて起動する場合などに有効である。また、アプリのバージョン情報を別ファイルに持たせておけばプロセス間通信を行わず、VASが当該ファイルを参照することによりチェックを行うことができる。

【0090】

さらに、アプリ起動後の実行時にもバージョンチェック処理を行うことができる。アプリ起動後の実行時には、プロセス間通信が使えるため、アプリの通常起動時におけるバージョンチェックと同じように適切なタイミングでバージョンチェックを行うことができる。

【0091】

また、アプリの実行時におけるバージョンチェックとして、例えば、アプリ側でAPIの引数の中にバージョン情報を入れておき、そのAPIをアプリが使用するときVAS140にAPIのバージョンを通知し、VAS140がそのバージョンをチェックするようにしてもよい。VAS140が、APIのバージョン差があると判断したときは、VAS140がそのAPIの要求をサービス層まで伝えないようにする。更に、アプリの実行処理を中止し、オペレーションパネル上にエラー表示を行って、ユーザにその旨を通知するようにしてもよい。バージョン差がない場合には、アプリの実行がそのまま継続して行われる。

【0092】

なお、バージョンチェック処理をVAS140側でなく、アプリ側で行ってもよい。その場合、アプリがVAS140からVAS140の有するAPIのバージョン情報を取得し、バージョンチェック処理を行う。そして、インストール時であればバージョン差があるか否かをVAS140に伝える。また、起動時、実行時であれば、バージョン差がなければ起動もしくは実行を継続し、バージョン差があれば、例えば、その旨のオペレーションパネル上に表示し、処理を終了す

る。

【0093】

このように、実施の形態1にかかる複合機では、VAS140が、アプリが使用するAPI単位毎のバージョン情報とVASのAPI単位毎のバージョン情報を取得し、対応するAPI同士のバージョンが同じか否かをチェックする。従って、アプリが使用していないVAS140のAPIが変更、追加、あるいは削除されて、VASのAPI全体(APIセット)のバージョンが変わっても、両者間の整合性に影響を与えないことを判断できる。全体バージョン同士を単純比較する場合と比べると、VASのバージョン差の吸収範囲が広がって、利用可能なアプリを増やすことができる。

【0094】

また、実施の形態1にかかる複合機では、VAS140が最初に全体バージョンを比較し、全体バージョンが異なっている場合のみAPI単位のバージョンチェックをすることができるので、バージョンチェック処理を効率良く、かつ迅速に行うことができる。また、予め必要なバージョン情報を持たせたアプリの実行ファイルを使うことにより、アプリの仮起動や通常起動の際のプロセス間通信により、VASが所望のバージョン情報を取得することができる。

【0095】

また、アプリのインストール時にバージョンチェックする場合、VAS140がアプリを仮起動させて、アプリが使用するAPI単位毎のバージョン情報を得ることができるので、通常起動をする場合と比較して効率的にバージョンチェックすることができる。

【0096】

(実施の形態2)

次に、実施の形態2について説明する。実施の形態1では、図4に示すように、アプリはアプリがVAS140に対して使用するAPIのバージョン情報を有し、VAS140はそのバージョン情報をチェックしていた。

【0097】

さて、複合機におけるVAS140は、図1に示すように全てのコントロール

サービスをカバーするのではなく、一部のコントロールサービスのみをカバーするように構成することもできる。実施の形態2では、VAS140を介さずに直接アプリと通信を行うコントロールサービスに対するAPIのバージョンもチェックできる構成について説明する。なお、以下の説明において、VAS、コントロールサービスを含めてそれぞれ“システム”と呼ぶ。

【0098】

実施の形態2では、図9に示すように、アプリは、VAS140に対して使用するAPIのバージョン情報の他、VAS140を介さずに通信を行うコントロールサービスに対して使用するAPIのバージョン情報を有している。VAS140は、アプリからこれらのバージョン情報を取得し、該当するシステムからそのシステムのAPIのバージョン情報を取得し、バージョン比較を行う。

【0099】

図9に示すように、アプリは、各システム毎にバージョン情報のテーブルを有していてもよいし、アプリが使用する全システムについてのバージョン情報を1つのテーブルに保持していてもよい。この場合、テーブル内でどのシステムのAPIかは識別できる。図9の場合、各テーブルは、API毎のバージョンに加えてAPIセットのバージョンも保持している。なお、アプリは、図10に示すような、システム毎のAPIセットのバージョンを保持したテーブルを持っていたりもよい。

【0100】

なお、図9に示す各バージョン情報は、アプリなどのプログラムの実行ファイル内に保持する他、別ファイルに備えておいてもよい。その場合、VAS140はプロセス間通信でなく、そのファイルを開くことによりバージョン情報を取得する。バージョン情報を保持するにはその他にも種々の方法が可能である。

【0101】

図11に、図9の構成における、VASによるバージョンチェック処理のフローチャートを示す。なお、図11のフローは、VASが、アプリからバージョン情報のテーブルを順次取得することを前提としている。また、インストール時のバージョンチェック処理を示している。

【0102】

まず、VASがアプリから1つのテーブルを取得する（ステップS701）。そして、そのテーブルがどのシステムに対応するものであるかを判断する（ステップS702）。次に、該当するシステムが有するバージョン情報のテーブルをそのシステムから取得する（ステップS703）。

【0103】

次に、アプリから取得したテーブルと、システムから取得したテーブルとを比較し、全バージョン（APIセットのバージョン）が一致するか否かを判断する（ステップS704）。一致する場合には、そのシステムに対してはバージョンが整合している旨を記録する（ステップS706）。そして、アプリ側から取得すべきテーブルが他に無ければバージョンチェック処理を終了する。他にあればステップS701からの処理を繰り返す。

【0104】

ステップS704にて、一致していない場合には、アプリがそのシステムに対して使用するAPI毎にバージョンをチェックする（ステップS705）。全APIのバージョンが同じであれば、そのシステムに対してはバージョンが整合している旨を記録する（ステップS706）。また、1つでもバージョンの異なるAPIがある場合には、バージョンが整合していない旨を記録する。このとき、どのAPIのバージョンが整合していないかを記録しておいてもよい。そして、アプリ側から取得すべきテーブルが他に無ければバージョンチェック処理を終了する。他にあればステップS701からの処理を繰り返す。

【0105】

バージョンチェック処理が終了した後、VAS140は、例えば、そのアプリが使用する全てのシステムに対してバージョンが一致していれば、そのアプリがインストール可能であることをオペレーションパネルに表示し、1つでもバージョンの不一致があるシステムがあれば、バージョンに不一致がある旨を表示する。バージョンに不一致がある旨を表示する際、どのシステムのどのAPIに不一致があるかを表示してもよい。

【0106】

本実施の形態により、VAS 140 に対する API のみならず、VAS 140 を介さずに通信を行う API のバージョンもチェックできる。

【0107】

(実施の形態 3)

実施の形態 1 にかかる複合機 100 は、VAS 140 が全アプリケーションに対して 1 つのみ存在するものであった。この実施の形態 2 にかかる複合機 900 では、各アプリごとに一つの VAS を起動し、各 VAS は対応するアプリとの間でバージョンチェック処理を行うようにする。

【0108】

図 12 は、実施の形態 3 にかかる複合機 900 の構成を示すブロック図である。図 12 に示すように、複合機 900 では、複数の仮想アプリケーションサービス (VAS) 941～948 がアプリケーション 130 の各アプリごとに動作している点が、実施の形態 1 にかかる複合機 100 と異なっている。

【0109】

VAS 941～948 は、プリンタアプリ 111、コピーアプリ 112、ファックスアプリ 113、スキャナアプリ 114、ネットファイルアプリ 115、工程検査アプリ 116、新規アプリ 117 および 118 に対応して、バージョンチェック処理を行う。

【0110】

図 13 は、実施の形態 3 にかかる複合機 900 の VAS 941～948 の構成と、VAS 941～948 と各アプリ、コントロールサービス層 150 および汎用 OS 121 との関係を示すブロック図である。なお、図 13 では、アプリケーション 130 として、プリンタアプリ 111、コピーアプリ 112、新規アプリ 117、118 の例を示し、さらにこれら各アプリに対応した VAS 941、942、947 および 948 を例として示しているが、他のアプリを使用する場合も同様の構成である。

【0111】

また、実施の形態 3 にかかる複合機 900 では、実施の形態 1 の複合機 100 と異なり、図 13 に示すように、各 VAS 941～948 と各アプリとの間には

VAS制御プロセス（デーモン）901が動作している。このVAS制御プロセス（デーモン）901は、VASが提供する全APIのバージョン情報を有している。

【0112】

仮想アプリケーションサービス（VAS）941～948のプロセスは、ディスプレイパッチャ145と、API単位バージョンチェックスレッド141と、全体バージョンチェックスレッド142と、バージョン情報取得スレッド143とが動作している。

【0113】

実施の形態3の複合機900における各スレッドの機能は、実施の形態1で説明したものと同様である。なお、スレッド141～143を1つのスレッドとしてもよい。

【0114】

また、VASの構成として上記の構成の他、図14（a）～（c）に示す構成をとることもできる。

【0115】

図14（a）は、各アプリケーションに対して起動されるVASを、親VASの子プロセスとする場合であり、親VAS自体は画面制御権（ユーザインターフェース）を持たない。図14（b）は、親VAS自体が画面制御権（ユーザインターフェース）を持つ場合である。図14（c）は、各アプリケーションに対応するVASの機能をスレッドとして起動する場合を示している。

【0116】

例えば、図14（c）の場合において、アプリが本起動していない時点では、アプリ対応のVASスレッドは起動されていない。アプリのインストール時に、アプリと対応付けられない1つのVASがアプリの仮起動を行い、バージョンチェックを行う。その後、アプリの本起動時に各アプリに対応したVASのスレッドが起動される。

【0117】

なお、本発明は、上記の実施の形態に限定されことなく、特許請求の範囲内

において、種々変更・応用が可能である。

【0118】

【発明の効果】

上記のように、本発明によれば、アプリが使用するAPI単位毎のバージョン情報とVAS等のシステムサービス側のAPI単位毎のバージョン情報を比較し、対応するAPI同士のバージョンが同じか否かをチェックできるので、アプリを正常に使用できるか否かを判断することができる。

【図面の簡単な説明】

【図1】

本発明の実施の形態1における複合機のソフトウェア構成を示すブロック図である。

【図2】

本発明の実施の形態1における複合機のハードウェア構成を示すブロック図である。

【図3】

本発明の実施の形態1における複合機のVASの構成と各アプリとコントロールサービス層と汎用OSとの関係を示すブロック図である。

【図4】

本発明の実施の形態1におけるバージョンチェック方法を説明するための図である。

【図5】

本発明の実施の形態1におけるバージョンチェック方法を説明するための図である。

【図6】

本発明の実施の形態1における仮起動可能なアプリのプログラム記述例である。

【図7】

アプリインストール時におけるバージョンチェックのシーケンスを説明するためのフローチャートである。

【図 8】

図 7 中のバージョンチェック処理を示すフローチャートである。

【図 9】

本発明の実施の形態 2 におけるバージョンチェックの方法を説明するための図である。

【図 10】

本発明の実施の形態 2 においてアプリが有するテーブルの例である。

【図 11】

本発明の実施の形態 2 におけるバージョンチェック処理の手順を示すフローチャートである。

【図 12】

本発明の実施の形態 3 における複合機のソフトウェア構成を示すブロック図である。

【図 13】

本発明の実施の形態 3 における複合機の V A S の構成と、V A S と各アプリ、コントロールサービス層および汎用 O S との関係を示すブロック図である。

【図 14】

本発明の実施の形態における V A S の構成例を示す図である。

【符号の説明】

- 100 複合機
- 101 白黒レーザープリンタ
- 102 カラーレーザープリンタ
- 103 ハードウェアリソース
- 110 ソフトウェア群
- 111 プリンタアプリ
- 112 コピーアプリ
- 113 ファックスアプリ
- 114 スキャナアプリ
- 115 ネットファイルアプリ

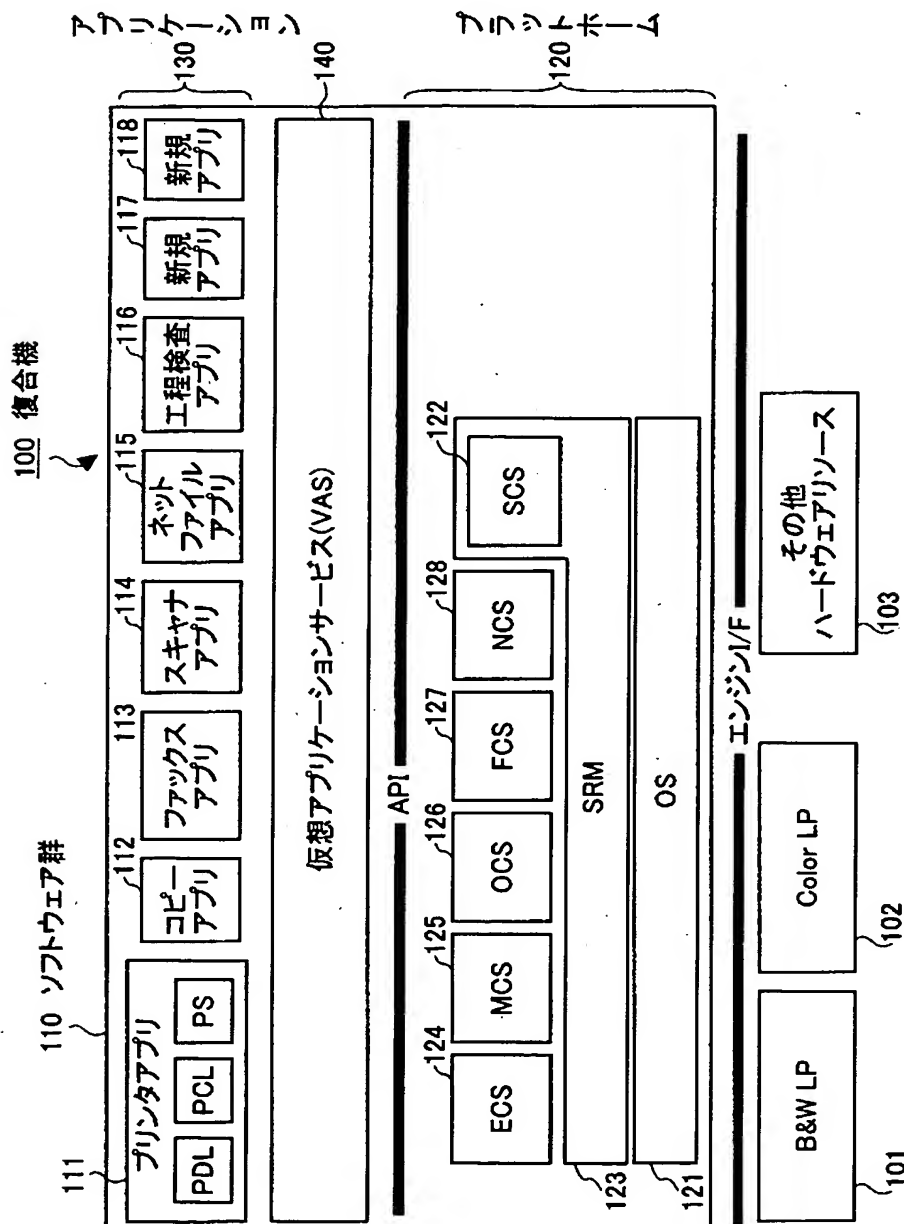
- 116 工程検査アプリ
- 117、118 新規アプリ
- 120 プラットホーム
- 121 汎用OS
- 122 SCS
- 123 SRM
- 124 ECS
- 125 MCS
- 126 OCS
- 127 FCS
- 128 NCS
- 130 アプリケーション
- 140、941～948 仮想アプリケーションサービス (VAS) 141
- API単位バージョンチェックスレッド
- 142 全体バージョンチェックスレッド
- 143 バージョン情報取得スレッド
- 144 制御スレッド
- 145 ディスパッチャ
- 150 コントロールサービス層
- 201 ラッピング処理情報ファイル
- 210 RAM
- 211 全APIテーブル
- 212 使用APIテーブル
- 900 複合機
- 901 VAS制御プロセス

【書類名】

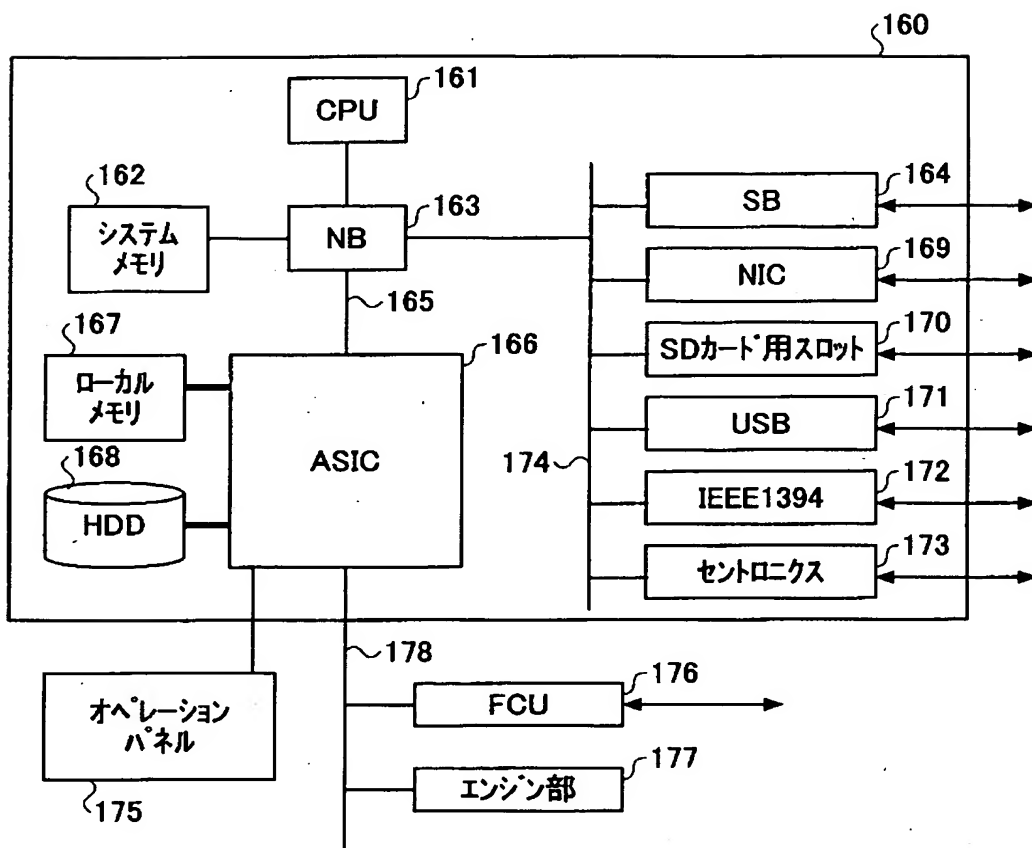
図面

【図 1】

本発明の実施の形態1における複合機の
ソフトウェア構成を示すブロック図

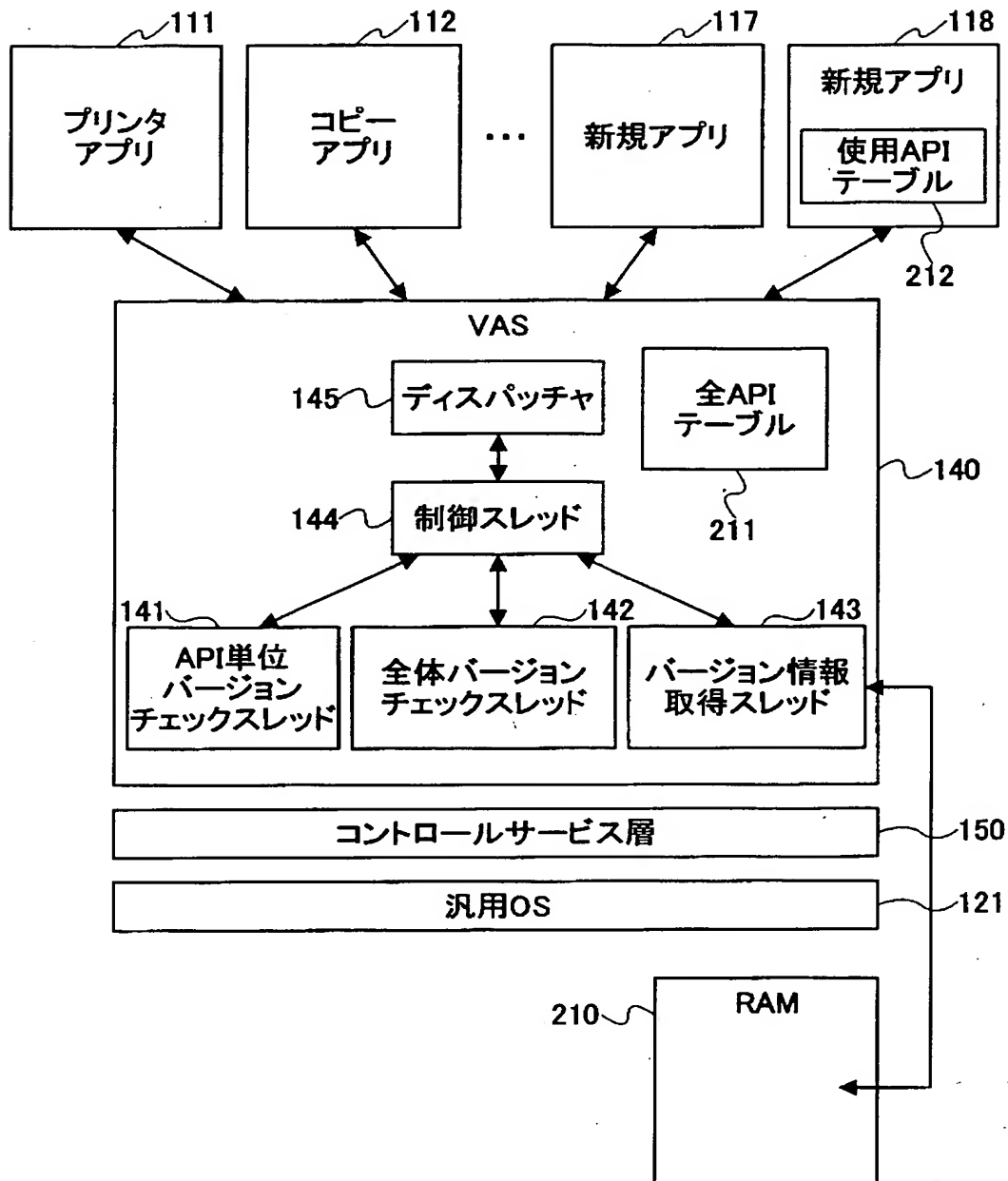


【図 2】

本発明の実施の形態1における複合機の
ハードウェア構成を示すブロック図

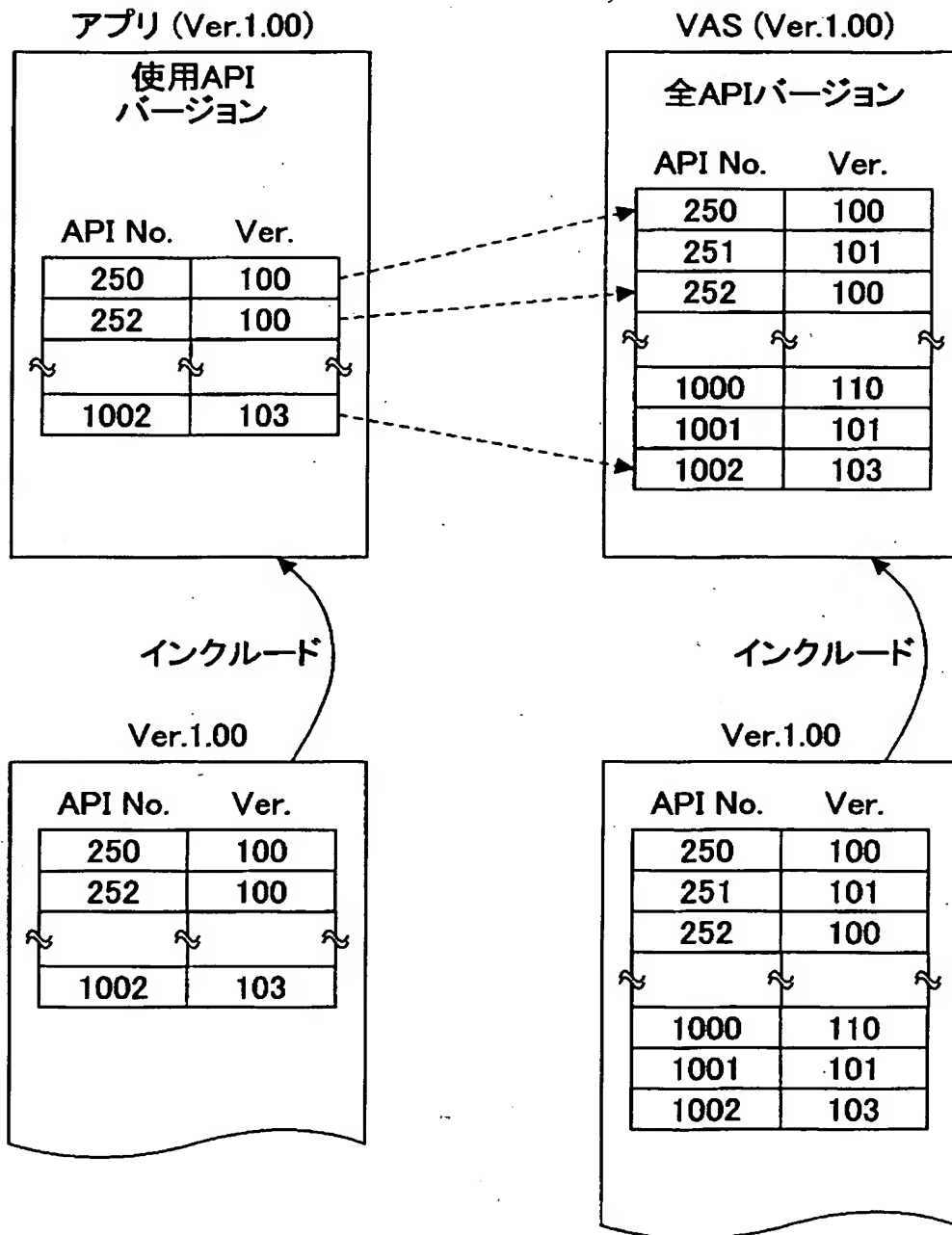
【図3】

本発明の実施の形態1における複合機のVASの構成と
各アプリとコントロールサービス層と
汎用OSとの関係を示すブロック図



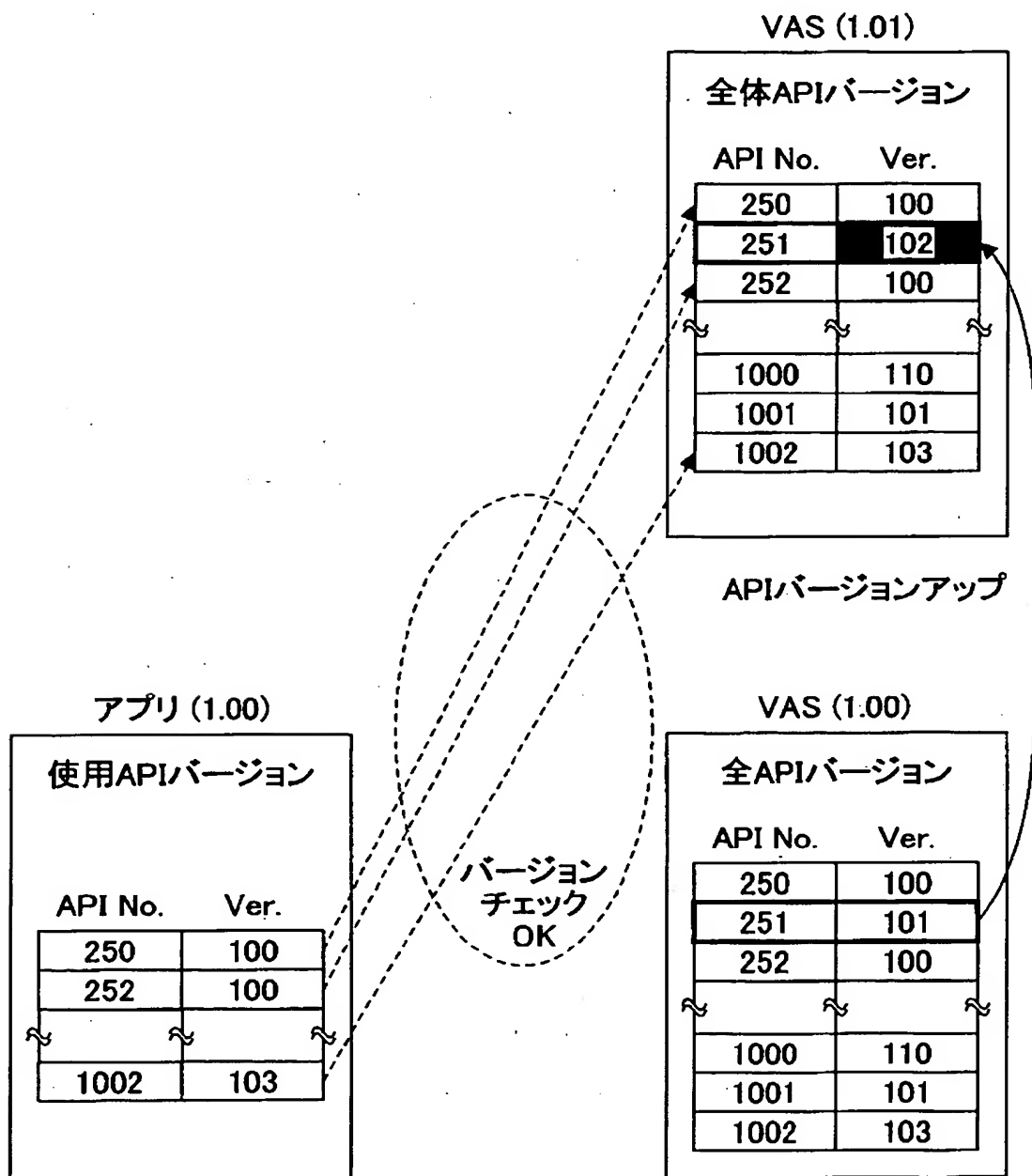
【図 4】

本発明の実施の形態1におけるバージョンチェック方法を説明するための図



【図 5】

本発明の実施の形態1におけるバージョンチェック方法を説明するための図



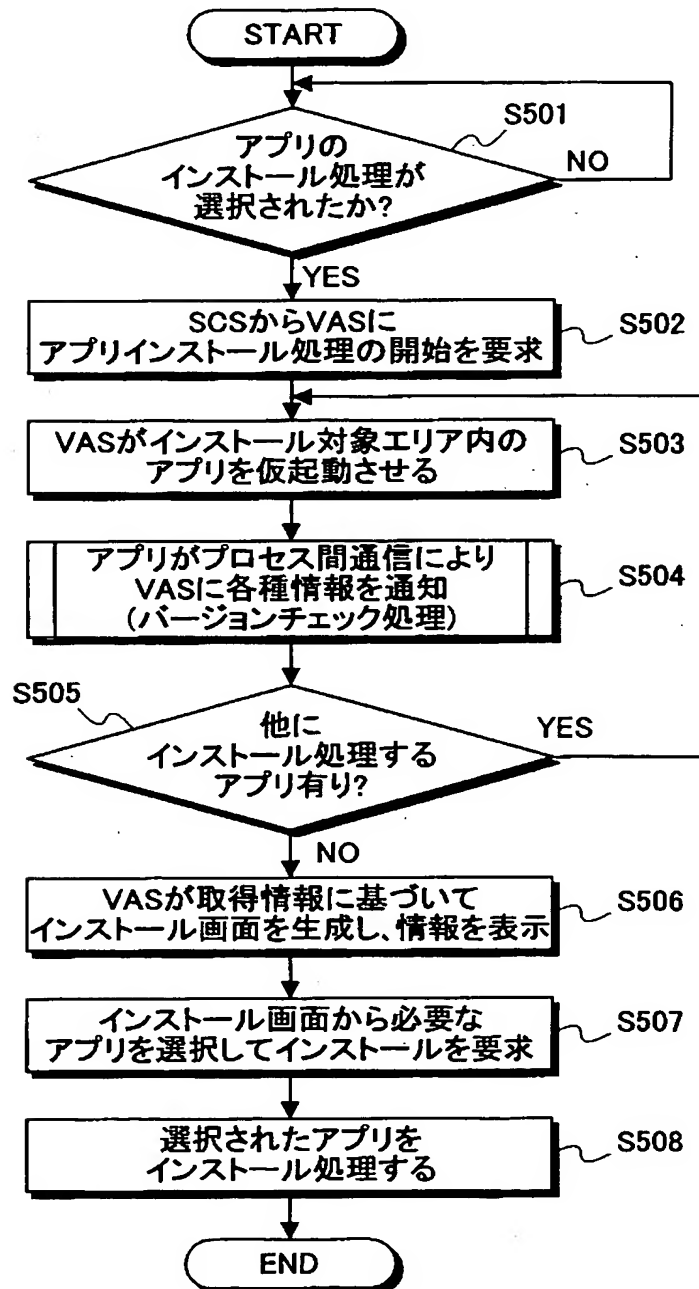
【図 6】

本発明の実施の形態1における
仮起動可能なアプリのプログラム記述例

```
int main(int argc, char**argv)
{
    if(argc==2) {
        if(strcmp(argv[1], "-v")==0)
            //仮起動実行
            if(ConnectVAS()==OK){ //VASとの通信準備
                SetApliInfo(); //アプリ情報提供処理
                Close //通信終了
            }
            exit(1)
        }
    }
    //通常起動処理
}
```

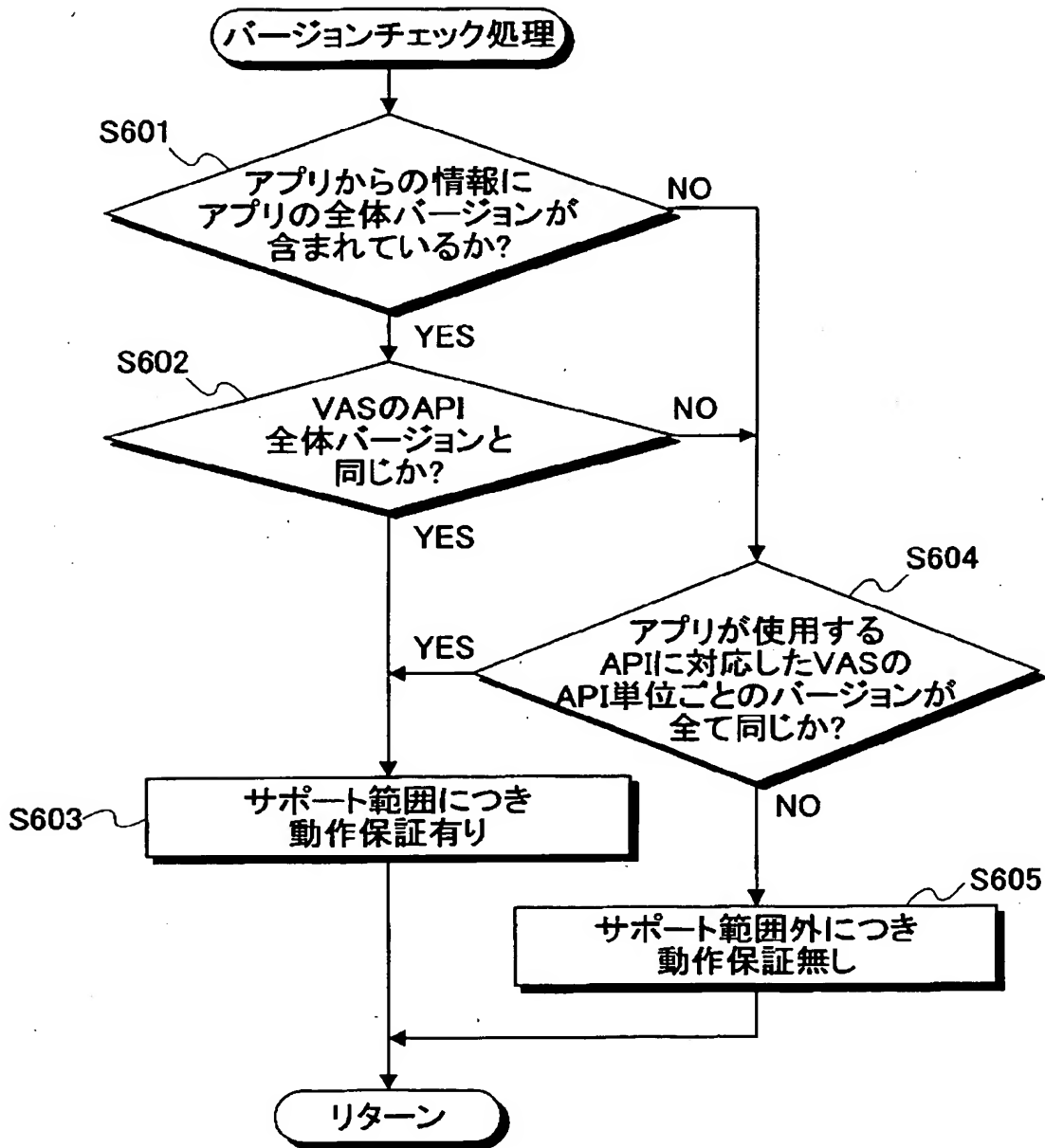
【図 7】

アプリインストール時におけるバージョンチェックの
シーケンスを説明するためのフローチャート



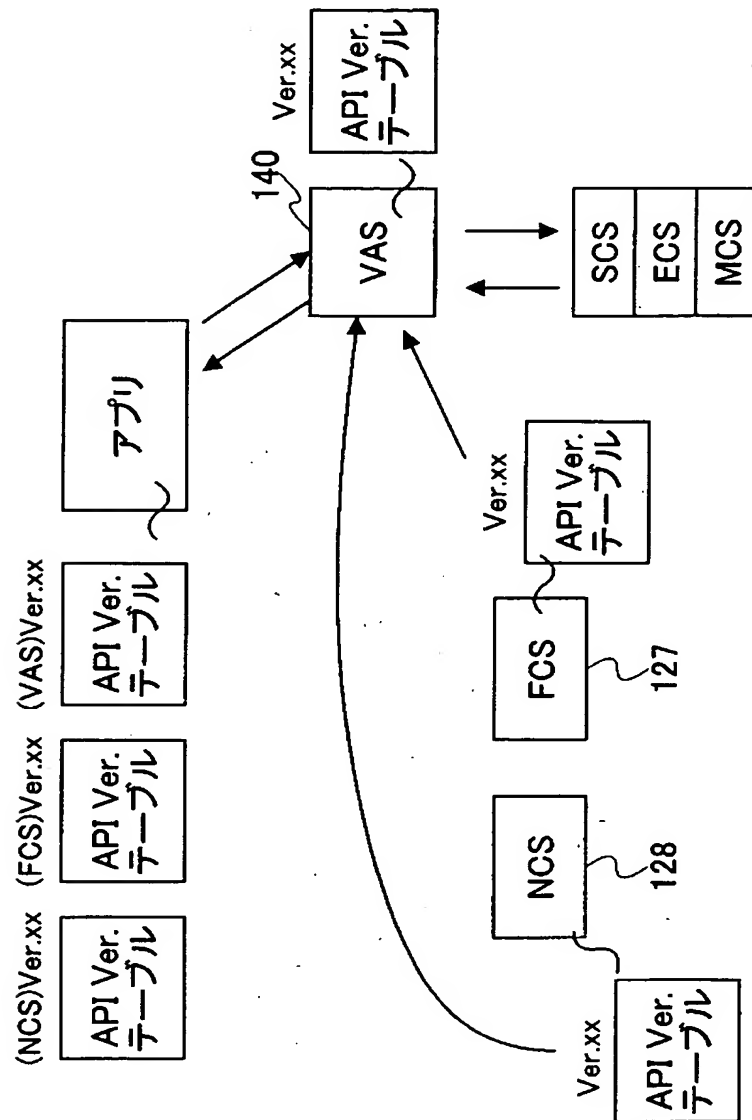
【図 8】

図7の中のバージョンチェック処理を示すフローチャート



【図 9】

本発明の実施の形態2におけるバージョンチェックの方法を説明するための図



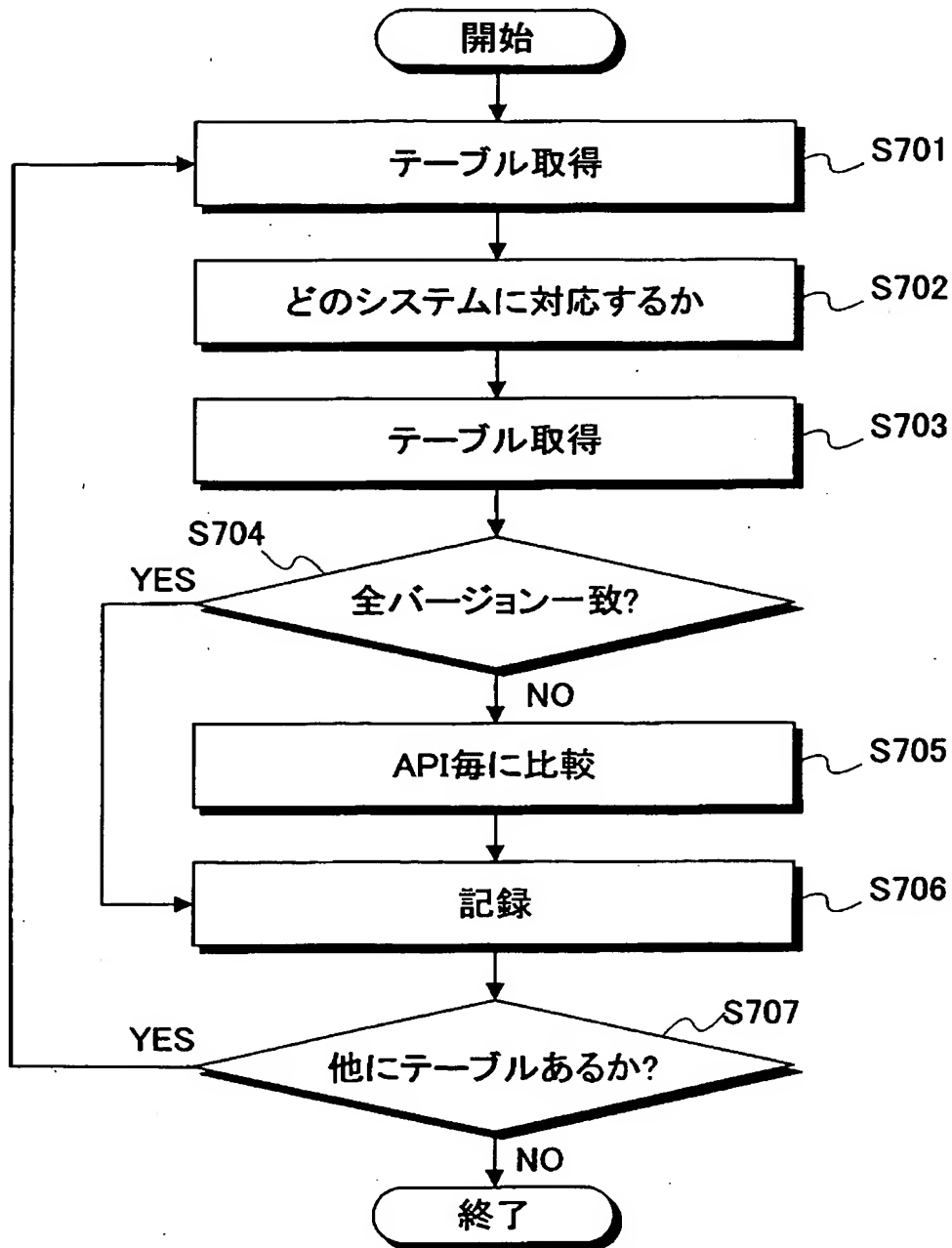
【図 1 0】

本発明の実施の形態2においてアプリが有するテーブルの例

NCS	Ver. 1.2
FCS	Ver. 2.5
VAS	Ver. 3.0

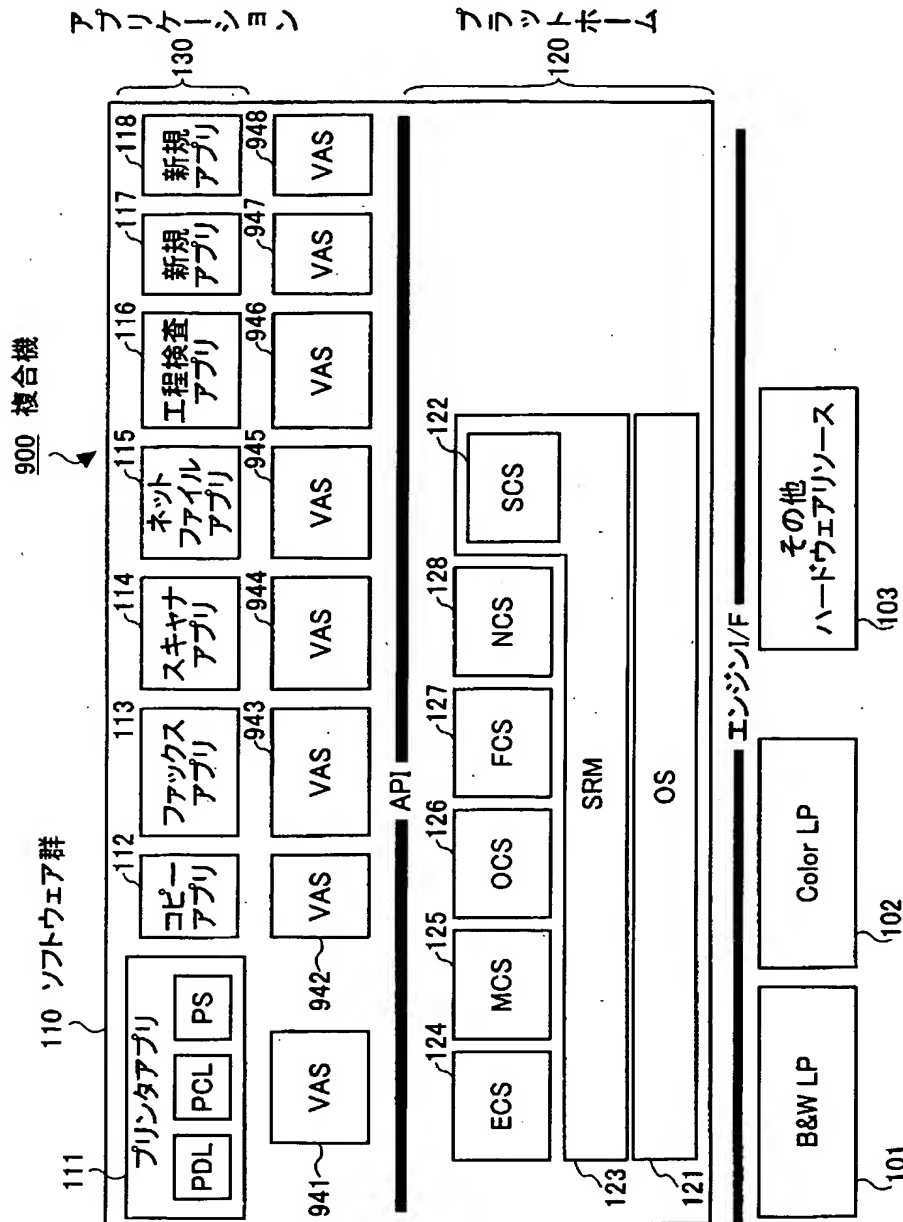
【図 11】

本発明の実施の形態2におけるバージョンチェック処理の
手順を示すフローチャート



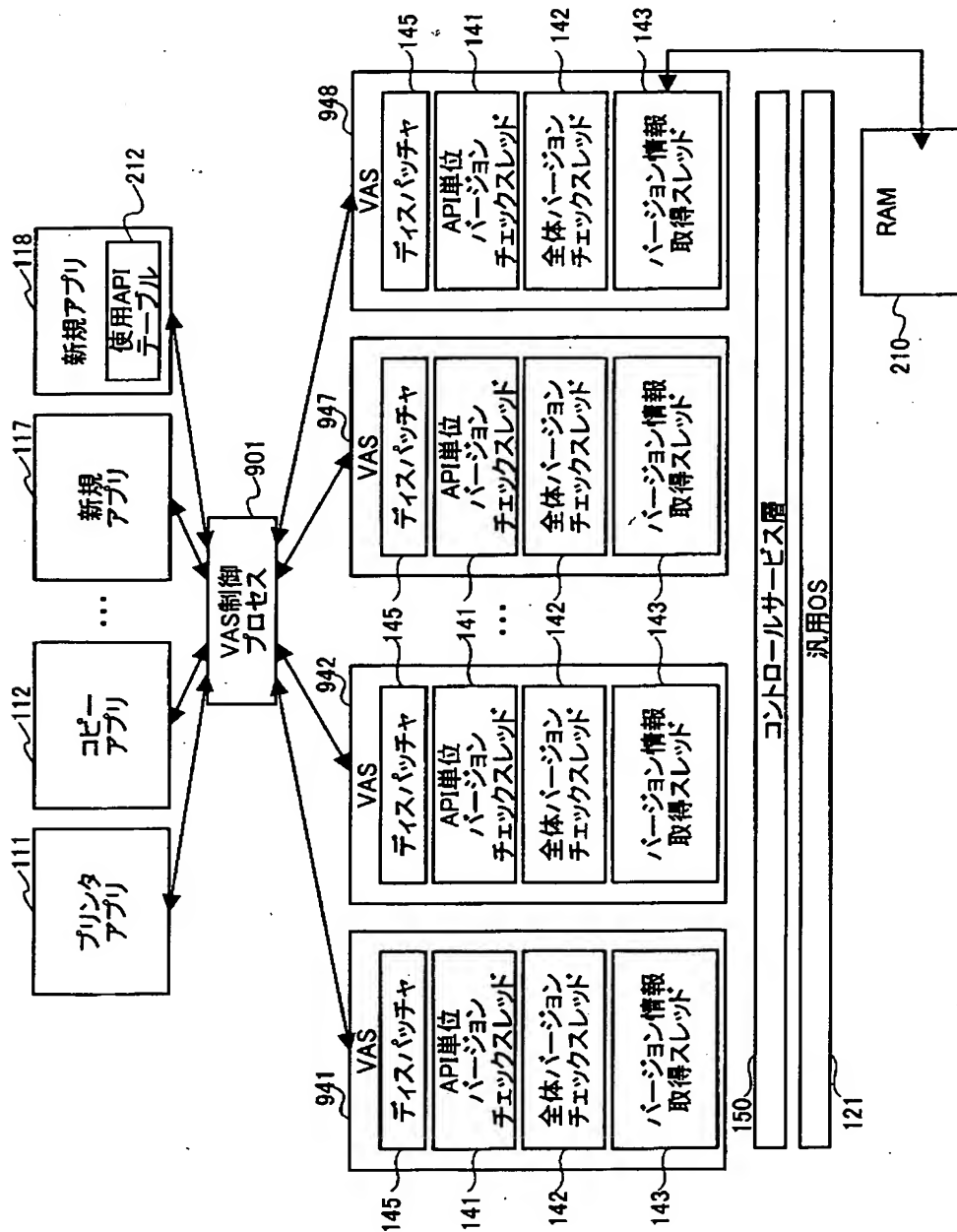
【図12】

本発明の実施の形態3における複合機の
ソフトウェア構成を示すブロック図



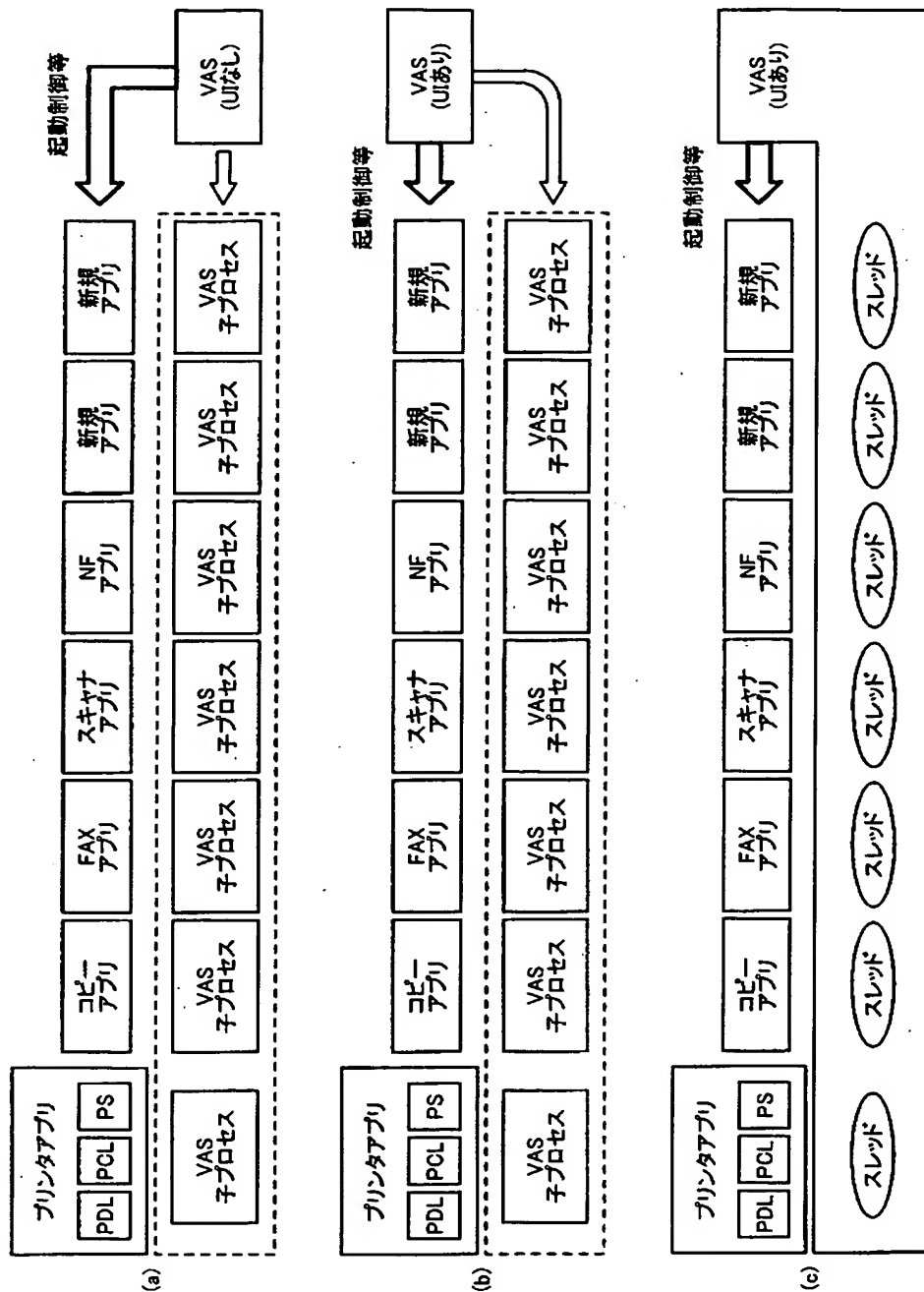
【図13】

本発明の実施の形態3における複合機のVASの構成と、
VASと各アプリ、コントロールサービス層および
汎用OSとの関係を示すブロック図



【図14】

本発明の実施の形態におけるVASの構成例を示す図



【書類名】 要約書

【要約】

【課題】 コントロールサービスなどのシステム側のサービスにおけるAPIのバージョンと、アプリケーションが使用するAPIのバージョンの違いをチェックする。

【解決手段】 アプリケーションと、前記アプリケーションからのAPIを用いた要求に基づきシステム側の処理を行うシステムサービスとを備えた情報処理装置に、前記アプリケーションがシステムサービスに対して使用するAPIのバージョン情報と、当該システムサービスが有するAPIのバージョン情報を取得する取得手段と、前記アプリケーションが前記システムサービスに対して使用するAPIのバージョンを、API単位に、前記システムサービスが有するAPIのバージョンと比較する比較手段とを備える。

【選択図】 図4

特願 2003-195194

出願人履歴情報

識別番号

[000006747]

1. 変更年月日 1990年 8月24日
[変更理由] 新規登録
住 所 東京都大田区中馬込1丁目3番6号
氏 名 株式会社リコー
2. 変更年月日 2002年 5月17日
[変更理由] 住所変更
住 所 東京都大田区中馬込1丁目3番6号
氏 名 株式会社リコー